



Nadando contra o Waterfall  
tail -f /mind/realworld >> blog

# 1up4Developers

sobre o blog e autores

80  
ago  
08

Por miguelhorlle  
Categorias: Uncategorized

Pessoal, estamos evoluindo (ou pelo menos tentando), como consequência disso realizamos o sonho do domínio próprio!

Agora quem quiser encontrar a galera do 1up4dev, deve acessar <http://1up4dev.org/>!

Bom pessoal, é isso, qualquer dúvida, críticas ou reclamações do novo layout ou qualquer outra coisa podem comentar aqui.

Obs.: Para quem assina o feed (RSS), já está sendo automaticamente redirecionado para o site novo.

Abraço

80  
jul  
08

Por miguelhorlle  
Categorias: interfaces, javascript e web  
Tags: save\_the\_developers



Bom pessoal, hoje estou aqui pra pedir ajuda. Sim, pedir ajuda pra todos àqueles que utilizam a web([www](http://www)) para alguma coisa, seja para estudar, trabalhar, jogar ou ficar rico! E falo por todos

desenvolvedores que tem o navegador como *container* de suas aplicações, que tem que lidar com as diversidades de dois mundos, MS Internet Explorer 6 contra a rapa!

O dia-a-dia no desenvolvimento de software web é um tanto sofrido para quem tem que, além de fazer uma aplicação com boa usabilidade, performance e ainda bonitinha, tem de manter compatibilidade com a ferramenta <voz\_do\_faustão>da gloriosa Microsoft</voz\_do\_faustão> que vem com o Windows XP: IE6. Sempre que você faz algum milagre em JavaScript, ou usa alguma técnica Web 2.0 pensando em agradar o usuário final (ou até mesmo o seu gerente!), tem que ficar com os dois pés atrás, pois você só pode comemorar depois de testar no IE6. O IE6 com certeza pode ser *superantecomp* navegador de todos os tempos, lento, péssima usabilidade, segurança terrível e ainda vai na contra-mão de todos os padrões WWW. O fato é que muita gente que programa não sabe que na verdade nem JavaScript este browser suporta, na verdade ele suporta um clone da Microsoft chamado JScript, este sim é o que deixa os programadores do mundo todo de cabelo em pé.

É claro que eu não podia deixar de fazer um comentário maldoso. Atualmente, ainda temos muitas aplicações que não usam muitos recursos maravilhosos que a nova geração de navegadores oferecem por culpa de empresas que não tem coragem de chutar o balde e forçar seus clientes à migrarem seus navegadores em prol de uma melhoria para todos, desenvolvedores e usuários! É claro que este tipo de empresa casualmente adotam metodologias duvidosas (waterfall puro!!), que refletem diretamente na qualidade de seus aplicativos, mas por outro lado temos empresas que lideram um movimento que deveria ser seguido por todos àqueles que querem se manter no mercado, uma delas é a [37Signals](http://37signals), que já manifestou algumas vezes o seu abandono ao IE6 dizendo que apartir do dia 15 de agosto todas suas aplicações passariam a não ter o compromisso de suportar o IE6.

Bom, acho que já ficou claro até aqui, que o passoal do 1Up4Developers apoia esta causa, e é por isso que escrevo este post, para mostrar a nossa aderência à campanha [SaveTheDelelopers](http://SaveTheDelelopers).

Assim como nós, muitas pessoas começam a apoiar essa campanha através de seus blogs, pessoas serotua

procurar arquivos do blog

IR

són-erbos-sianabias

Sobre o Blog e Autores

deef SSR

Assine o feed!



61023arbutuo

S T Q Q S S D

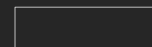
1 2 3  
4 5 6 7 8 9 10  
11 12 13 14 15 16 17  
18 19 20 21 22 23 24  
25 26 27 28 29 30 31

< ago

sa i rogetac

banco de dados  
hello  
interfaces  
javascript  
metodologias  
orm  
pragmatic waterfall  
processos  
resenhas  
Uncategorized  
web

retnuoc deef



stats gol b

968 hits

estatisticas

Evolução!  
Pensando na gente (desenvolvedores!)  
Resenha do livro Pragmatic Unit Testing  
Aderindo a Campanha BR-Linux  
A importância de estudar constantemente

estatisticas

Nenhuma

so i rátnuoc


miguelhorlle em Pensando na gente (desenvolvedores!)  
rodrigopanachi em Pensando na gente (desenvolvedores!)  
Daniele em Aderindo a Campanha BR-Linux  
miguelhorlle em A importância de estudar constantemente  
Roger Leite em A importância de estudar constantemente


que são referência no desenvolvimento de software web. Eu descobri esta campanha através do blog Nome do Jogo, do Carlos Brando, e de imediato pensei em fazer este post.


Pessoal, disseminem esta idéia em todos os lugares, quando algum parente chamar você para consertar seu computador diga: só se você atualizar o seu navegador!


Abraço

02  
jun  
08

 Por Roger Leite

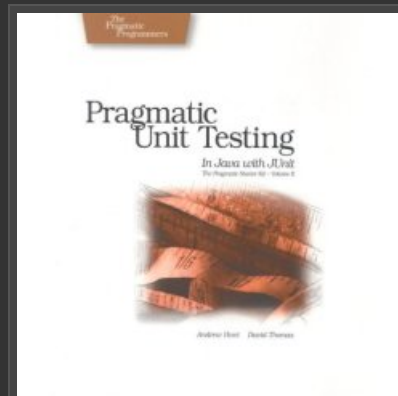
 Leave a Comentário

 Categorias: pragmatic waterfall e resenhas

 Tags: resenhas, testes unitários

Olá a todos !

Sei que estão estranhando ... perceberam que não tem Waterfall no título ? Pois bem, hoje não vou chorar e digo mais, vou até fingir que não vivo num *Waterfall* e vou falar sobre TESTES ! Tá bom, sei que enfatizei demais, só vou fazer uma resenha sobre este último livro que li mesmo.



Numa leitura leve e até divertida (sou nerd mesmo), os autores abordam conceitos práticos de testes que não estão ligados diretamente ao JUnit, e sim a “Filosofia de Testes”. O legal que os principais conceitos são apresentados com acrônimos como “Right BICEP”, “CORRECT Boundary Conditions”, “A TRIP”, MockObjects e etc. Depois da passagem por todos esses acrônimos, os próximos capítulos atacam temas como, onde colocar os testes, design dos testes e etc.

Isso pode parecer estranho, mas de todos os capítulos o que eu mais gostei foi do primeiro, a Introdução, talvez porque no momento estou com a água do waterfall até o pescoço, e nele os autores colocam as dicas de como contra-argumentar as

desculpas para não fazer testes. Exemplos dos tópicos, “Por que devo me importar com testes ?” e “Desculpas para não testar”, parece que os autores realmente conhecem o lado negro da força. Por sinal, achei este último tão interessante, que estou pensando em pedir permissão para traduzi-lo e postar aqui, se alguém souber o caminho das pedras e quiser ajudar eu peço a gentileza de entrar em contato.


Gostei muito do livro, o considero uma ótima referência sobre o tema, veja bem, **referência**, pois se queres uma biblia do JUnit, descarte-o. Sei que muitos da nossa área não conhecem nada sobre o assunto, e um ótimo começo seria por ele.


Agora, voltando um pouco pra minha (e de muitos) realidade cruel, antes de ler o livro eu imaginava (ou sonhava ?) que o sistema atual em que trabalho, poderia ser implantado testes, agora, com uma visão mais pragmática, tenho certeza que estava certo, só que mirando na camada errada. Aqui, a maioria da lógica (uns 90%) está em PL/SQL no banco, e a melhor maneira de implantar testes seria começando com um PL/SQLUnit ... mas aí já é assunto pra outro post. Ahh, ainda não pesquisei, mas deve existir com certeza.


Chegando ao fim do livro ...


Uma parte chata do livro foi quando terminei de lê-lo, confesso que fiquei com uma vontade de “quero mais” e acabei ficando com a impressão de que só li a ponta do iceberg sobre o tema. Sugestões de mais livros sobre o tema, são bem vindas !

02  
jun  
08

 Por Roger Leite

 1 Comentário

 Categorias: hello

 Tags: campanhas

Não é só de Waterfall que este blog vive. Ele também nasceu pra ajudar o software livre ! Eu sei, não temos nenhum post sobre isso, mas isso é questão de tempo eu prometo.

Como grande fã que sou do BR-Linux, estou aderindo a campanha ! E com certeza todos os autores deste blog concordam comigo, que notícia do mundo livre é no BR-Linux.

**Ajude a sustentar a Wikipédia e outros projetos, sem colocar a mão no bolso, e concorra a**



miguelhorle



Roger Leite



Humberto B



rodrigopanachi

ved45 j

Linux Systems Administrator em Google  
Backend Engineer em inDinero  
Lead Frontend Engineer em inDinero  
Gerente de Projetos: em LBSLocal  
(Apontador & MapLink)  
Web Development Software Engineer-  
NYC em FactSet Research Systems  
Software Engineer - NLP Parsing em  
FactSet Research Systems  
Senior Front End Developer em Everyday  
Health Inc.  
Skilled Object Oriented Programmers em  
Crowd Fusion  
Web Developer em Ci&T  
Desenvolvedor Java Web em Pinuts  
Studios  
Desenvolvedor Java Mobile  
(Android/BlackBerry) em Pinuts Studios  
Estágio Desenvolvedor Redes em EngSys  
ESPECIALISTA EM PROCESSAMENTO  
DIGITAL DE SINAIS - JÚNIOR em Vocalize  
DESENVOLVEDOR DE SOFTWARE -  
JÚNIOR em Vocalize  
ESPECIALISTA EM PROCESSAMENTO DE  
LINGUAGEM NATURAL - PLENO em  
Vocalize

um Eee PC!

...e também a pen drives, card drives, camisetas geeks, livros e mais! O [BR-Linux](#) e o [Efetividade](#) lançaram uma [campanha](#) para ajudar a Wikimedia Foundation e outros mantenedores de projetos que usamos no dia-a-dia on-line. Se você puder doar diretamente, ou contribuir de outra forma, são sempre melhores opções. Mas se não puder, veja as regras da promoção e [participe](#) – quanto mais divulgação, maior será a doação do BR-Linux e do Efetividade, e você ainda concorre a diversos brindes!

desenvolvimento  
jun  
08

Por [rodrigopanachi](#) 2 Comentários

Categorias: processos  
Tags: agile, poo, testes unitários

Já faz tempo que venho ensaiando este post. Minha idéia é mostrar como é importante na nossa profissão de “desenvolvedor” estar constantemente aprendendo novas técnicas, linguagens, frameworks, metodologia, etc. Com um mercado tão competitivo como o de Desenvolvimento de Software, não podemos nos dar o luxo de conhecer apenas uma linguagem. Evidente que é bom que você escolha uma para se especializar, mas de forma alguma deve ser a última linguagem que você aprenderá.

Há algum tempo atrás, orientação a objetos era uma coisa de outro mundo para mim. É sério, não conseguia pensar na possibilidade de existir outro paradigma de programação. Bem, depois que estudei muito sobre OO e passei a utilizar profissionalmente, hoje não consigo me imaginar trabalhando sem OO. Tudo fica claro, organizado, abstraído... O que eu ganhei com isso? Com certeza consegui ser mais produtivo, mais organizado e mais eficiente e, como consequência, melhor remunerado. Se ainda desenvolvesse proceduralmente, certamente ainda desconheceria conceitos e técnicas, sendo apenas mais um na multidão. E não é assim que um verdadeiro desenvolvedor ágil quer ser visto, certo?

Outra coisa que estudei muito e hoje fico feliz em utilizar profissionalmente são Testes Unitários. Uma das premissas do desenvolvimento ágil está relacionada à qualidade do código. Com testes unitários é possível desenvolver incrementalmente e responder rápido às mudanças pois seu código está “protegido”. Além de servir como apoio à refactoring. O que eu ganho com isso? Consigo me preocupar apenas com o desenvolvimento de uma pequena funcionalidade por vez, meu código fica mais “limpo” e manutenível. Em consequência disto me torno mais ágil e sou melhor remunerado. Além de poder dormir mais tranquilo...

É importante reconhecer que há muito para aprender ainda. Nosso cenário de trabalho muda constantemente e os “usuários” são cada vez mais exigentes. Além disso, é importante lembrar que [não existe bala de prata](#), não existe uma tecnologia que resolve todos os problemas. As linguagens e tecnologias são limitadas e precisam evoluir. Você precisa evoluir junto! Não se esqueça também que lá fora tem um mercado de trabalho começando a enxergar essas qualidades ágeis.

desenvolvimento  
jun  
08

Por [Humberto B](#) Leave a Comentário

Categorias: metodologias e processos  
Tags: devaneios, guerrilha

Eu preciso desenvolver uma idéia que vem me provocando ultimamente. Na verdade é menos uma idéia do que um reflexo da situação desoladora em que a maioria de nós está.

Em se tratando de 1up4dev, nem preciso dizer que a situação a que me refiro é a de quase inevitabilidade do waterfall, em que estamos tão engolidos pelo “sistema” que, aparentemente, só nos resta lamentar, se frustrar e eventualmente se acostumar com tudo. Não deduzo, porém, que esses são estágios de uma manifestação de bunda-molice. Do contrário, seria suficiente encerrar o assunto com algo do tipo “*ou somos parte da solução ou do problema*” (doutrina Bush, em pleno 2008). E ainda assim, sem que ao menos sejam esboçados tanto “o problema” como “a solução”, esse raciocínio binário não serviria para nada.

Antes de continuar com a minha idéia, gostaria de escrever rapidamente sobre o panorama que se desenha para o futuro. Especificamente, o nosso futuro. Nem sempre lembramos dele, mas é lá que vamos viver em breve. E sem querer parecer auto-ajudesco, digo que o futuro do desenvolvimento de software está nas nossas mãos — e não de forma indireta ou abstrata. É lógico: as mãos que hoje controlam o “sistema” vão se aposentar daqui uns anos, e as nossas vão substituí-las. Nessa sequência, é possível imaginar que, em breve, estaremos perpetuando o waterfall. Pois nesse dia nós é que seremos o status quo, e o status quo, para ser digno do nome, não quer saber de mudar nada.

Software já é estratégico há algum tempo e ocupará cada vez mais espaço na vida das pessoas, das empresas e dos governos. Que qualidade de software será oferecida quando nossa geração estiver no comando? Imagine o alto custo financeiro e social de se manter na periferia de TI. Sub-desenvolvimento passa a ter um novo significado, não é? Temos que assumir a responsabilidade, até porque ela pode significar a existência dos nossos empregos. Ou você vai preferir usar um software made in India?

Nada contra a Índia, claro. Mas o alerta já tinha sido dado por David Anderson no começo do livro Agile Management for Software Engineering. Abaixo traduzo livremente um trecho cujo original você encontra na página xxvi do livro:

*Se a atividade intelectual de software tiver que permanecer nos países desenvolvidos e se os engenheiros de software americanos, europeus e japoneses quiserem manter o alto padrão de vida ao qual se acostumaram, eles devem aumentar sua competitividade. Há um mercado global de desenvolvimento de software, o que encolheu a distância entre um cliente na América do Norte e um fornecedor na China ou na Índia.*

Esse medo do sr Anderson tem que ser nosso também (exceto que nós não temos como rebaixar nosso “alto padrão de vida”). Sabemos que precisamos melhorar, e muito, a nossa competitividade, e não só individualmente. Agora, a grande questão é: como vamos fazer isso, se não temos suporte para viabilizar novas formas de trabalho sem dispararmos o sinal de pânico no chefe, no cliente?

Pretendo desenvolver a minha sugestão em alguns posts, seguindo alguns princípios:

1. “Mudança” é definida como o amplo abandono da mentalidade waterfall no mercado de TI.
2. Você acredita na mudança e é o maior interessado nela, pois ela representa o futuro que **você** quer.
3. O risco da mudança é percebido com mais intensidade quanto maior é o poder de quem a observa.
4. O benefício da mudança é percebido com menor intensidade quanto maior é o poder de quem a observa.
5. A mudança pode ocorrer de baixo para cima na cadeia de poder.

No próximo post, pretenderei detalhar melhor os efeitos desses cinco pontos. Dali em diante, dificilmente irei sugerir uma ação coordenada e planejada — nada menos ágil que isso! Prefiro apostar no desenrolar natural das coisas, desde que os princípios sejam válidos. No mínimo, vamos avançar o debate. Tomara que eu não escreva muita besteira, e espero ser corrigido a qualquer momento.

Assespro  
jun  
08

Por Roger Leite 4 Comentários

Categorias: metodologias e processos

Tags: devaneios, paranoia

Falta pessoal qualificado em TI, diz Assespro

<http://www.guj.com.br/posts/list/15/92783.java#497015>

Esta discussão está boa, e no meio das chamadas levei um “impacto” ao ler a frase abaixo.

**louds wrote:** [...] Não usem a incompetência alheia como desculpa. Se está ruim e você não está ativamente combatendo isso, você é parte do problema e não da solução. [...]

Esta é uma das frases que eu deveria lembrar toda vez que começo a lamentar sobre o ambiente atual de trabalho, eu tento, me gerencio, mas confesso que é difícil.

Com o diabinho no ombro, logo ouvi um suspiro ... Waterfall é bom ! Não questione ... volte a codificar ... Só que ao olhar pro lado, vi o meu poster do Dijkstra e voltei a realidade. Foi quando me perguntei:

**Por que é difícil combater o Waterfall !?!**

Sei que esta é uma pergunta sem resposta direta e que depende de muitos fatores do seu ambiente de trabalho, mas eu numa análise fria e cruel respondo: **Porque a maioria não sabe o que é waterfall.**

No último ano da faculdade (cursei Sistemas de Informação), e sem brincadeira, nas aulas de Engenharia fui obrigado a decorar todas as fases, papéis, artefatos e etc. do RUP, e pra fechar com chave de ouro, no segundo semestre veio o famoso Análise de Ponto de Função. Isso somente prova que já aprendemos errado, e sei que muita gente fica nisso. Se hoje tivesse a oportunidade de

encontrar este mesmo professor, iria fazer a pergunta acima, e acrescentar com questionamentos como, engenharia de software não é eng. civil e neste processo todo quem está preocupado com o Retorno de Investimento do cliente !?!

Acredito que a chave para combater o Waterfall é conhecimento, temos que trazer questionamentos a quem está do lado e isso não é nada fácil, como você convence os mais de vinte desenvolvedores que estão ao seu redor que eles estão na Matrix !?! O que fazer com o pessoal que não quer sair da Matrix !?! **O Waterfall é uma grande mãe que coloca muitas pessoas (Analistas, Desenvolvedores, Gerentes, Testers ... etc.) numa casca irreal de proteção e que gera um ciclo vicioso que se auto sustenta.**

Neste exato momento que sou obrigado a concordar com o grande malucão Raulzito:

*É pena não ser burro ! Não sofria tanto.*

As vezes me questiono sobre esta insistência em tentar mudar, nadar contra o Waterfall. Não seria melhor se “render” logo e entrar no jogo, sei lá, poder falar, “Caso de Uso não é comigo, só programação!” e ver que são 10 pras 18, desligar o computador e sair sem o mínimo peso na consciência.

Por que numa equipe de dez desenvolvedores, só eu sinto falta de testes unitários !?! Fico indignado ao receber um caso de uso que não agrega nada ao cliente, somente foi “inventado” pra ser cobrado dele, e só eu que vejo isso !?! Sei lá, eu devo estar maluco mesmo com toda esta historia de desenvolvimento ágil, e se preocupar com ROI do cliente, apesar de achar uns malucos por ai que sofrem do mesmo “problema” que o meu.

Como no filme dos 300, tenho esperanças e sei que poucos enfrentarão muitos, e continuo minha batalha com o Waterfall.

Paz a Todos !

Obs: Sei que o RUP é mal usado, pois uma brecha dele é ser muito genérico, e isto cada vez se confirma mais, que o modelo atual “tradicional” está falido.

babar  
jun  
08

Por Roger Leite    Leave a Comentário  
Categorias: pragmatic waterfall  
Tags: colaborativo, dilbert, guia de sobrevivência



Tradução rápida do diálogo:

Chefe (com chifrinhos): Por que você levou seis meses para completar esta simples tarefa ?

Dilbert: Por causa das suas mudanças contínuas, sua comunicação confusa e seu pequeno expediente de trabalho.

Chefe (com chifrinhos): Estou procurando por alguma coisa mais parecida como você sendo preguiçoso.

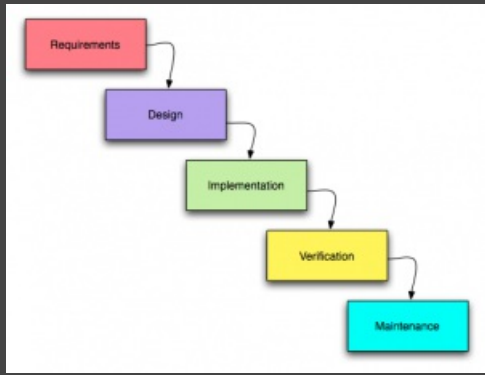
Hello hello hello ! Estas tirinhas do Dilbert estão de matar ultimamente. E já pra avisar, o TPW significa *The Pragmatic Waterfall*, um novo termo para o que buscamos aqui neste blog, ajudar quem sofre com o *Waterfall* ! Ok, isso inclui a mim mesmo.

Quem vive num ~~mal~~ indigno ambiente Waterfall já deve ter vivenciado muito disso que ocorreu acima, o gerente “junior” procurando uma desculpa de porque o gant chart está vermelho para repassar ao gerente “pleno” que este repassará ao “senior junior” e que este repassará “senior pleno” ... bom, já entenderam até onde a desculpa vai chegar.

Na tentativa de transformar este post de “muro das lamentações” para Pragmatic Waterfall, vamos as dicas didáticas (ou seria um guia de sobrevivência?) de como tentar contornar este tipo de situação frustrante:



==>>



- Gerador de código descartável. Sim, é isso mesmo que você leu, o gerador de código você já sabe o que é, agora o descartável é o que você deve estar imaginando. Isso não tem muito a ver com o Waterfall, mas todo projeto que trabalhei tem aquelas camadas que repassam chamadas e seguem um padrão comum. Logo, você não precisa – e nem deve – escrevê-los, para isto inventaram o computador. Se você tem sorte de usar um sistema unix like, se aventure com shell scripts, vale a pena, caso não tenha esta sorte ... err, primeiro sinto muito por ti ... mas você tem opção de linguagens de scripts como Perl, Python e Ruby em suas mãos, aproveite ! Crie estes scripts descartáveis e gere toneladas de código, seu chefe vai ficar feliz da vida com o aumento da produtividade.
- Conheça todos os recursos que a sua IDE ou seu editor de texto oferecem. Isso parece ser uma dica besta, mas pode acreditar que não é, já vi muita gente usando um mesmo editor por mais de meses e ficam “abobados” ao descobrirem que o Ctrl+H abre a tela de Replace !!! Bom no meu caso que uso Eclipse o dia todo, as dicas são:
  - Aprenda a usar teclas de atalho, elas realmente aumentam a produtividade. *Cheat Sheets* como [este](#), ajudam no processo de adaptação.
  - Use e abuse dos Templates, aquelas configurações chatas de xml que toda hora são necessárias, não perca tempo, crie um template para isso e seja feliz. Você pode criar também para aqueles métodos chatos com assinaturas iguais sempre (tipo do Struts mesmo sabe !?) ... o céu é o limite !
  - Aprenda a usar as opções do menu Refactor e – adivinhe! – Geradores de Códigos.
- Mantenha um *checklist* de documentos a atualizar, aqueles do tipo, Requisitos, Casos de Uso, Arquitetura, Alteração, Instalação ... etc. Com um *checklist* você não precisa ocupar a cabeça com este passo muito importante do *waterfall* e lembre-se que neste ambiente o que é valorizado são os documentos e não as pessoas.
- Antes de começar a sua nova tarefa que está no Gantt, descubra quem são os envolvidos, neste nosso ambiente temos especialistas para todos os lados, converse com eles e feche pactos, pois é muito comum no final da tarefa você descobrir que a procedure retornará um tipo complexo e não um cursor como você imaginava.
- Sei que isso pode parecer irreal para você que está num *waterfall* enraizado, porém, tente pelo menos. Tenha um ambiente mock para desenvolvimento, isso pode te salvar ao manter sua tarefa “verdinha” no Gantt Chart. Sim, todos nos sabemos que o Gantt Chart não funciona, porém eu e você que estamos no *waterfall* temos que usar e até fingir que funciona.

Espero que com essas dicas você consiga se livrar de suas tarefas em menos tempo, e com o tempo que sobrar, aproveite para aprender coisas novas, metodologias novas e descobrir que existe vida fora do *waterfall* ... e acredite ! Estão documentando, aqui, aqui e aqui !

Então, nobres colegas e respeitável público, o nosso blog estava meio estranho no Blogger e decidimos mudar para uma plataforma mais funcional, mais gerenciável, mais colaborativa, enfim, uma coisa totalmente mais 2.0 mesmo.

O conteúdo continua composto de puro veneno partindo de mentes furiosas por não terem uma vida agilmente ativa. Aposto que você, leitor, também se encontra nessa situação aviltante, então assine o novo feed e participe comentando as nossas observações, frustrações e viagens.

Muito sucesso a todos! Prossigamos...

Primeiramente tenho que admitir que o termo usado no título do post não existe em nenhum dicionário convencional (encontrei apenas no UrbanDictionary, mas não era bem o que queria expressar...*risos*). Mas mesmo assim vou usá-lo livremente, pois acho que não teria nenhum verbete mais adequado para expressar como me sinto atualmente (profissionalmente falando claro!), nada descreve o fenômeno que é desenvolver software, ou pelo menos tentar, em um mercado onde praticamente 99% das grandes empresas ainda gastam milhares de reais com consultorias *especializadas* em implementar metodologias e processos que no fundo só servem para gastar tempo, dinheiro e a paciência dos colaboradores envolvidos. O resultado disso é uma empresa certificada (CMM/i, MPS.br e afins) e dezenas de funcionários *estressados*.

É impressionante como a falsa segurança de um processo todo controlado, medido e previsível (isso é o que os *chairmen* ainda pensam!) ainda está presente nos gestores de TI atuais, pelo menos no Brasil.

O Waterfall continua enraizado em nossa cultura de gestão por simples jogo de interesses. Essas metodologias (CMM/i e similares...) só beneficiam pessoas que não querem se comprometer, não estão interessadas na real satisfação do cliente e querem se manter no mercado, muitas vezes sendo incompetentes no que fazem (afinal este tipo de processo permite que as pessoas se escondam atrás desta burocracia). Existem milhares de papéis (analista, projetista, analista de negócios, gerentes e mais gerentes, analista de qualidade...blah blah blah) a serem desempenhados, mas estes papéis são tratados como se fossem exercidos por robôs. Isso gera o tipo de frase: "Mas eu faço análise, prazo não é comigo!".

Eu vejo este tipo de metodologia como a velha discussão dos sistemas sócio-políticos. Se analisarmos de forma fria e racional as duas principais vertentes desenvolvidas neste campo, percebemos que de um lado temos o socialismo com todo seu esforço para ser algo justo e equilibrado, e na outra ponta temos o capitalismo com toda sua desigualdade, agressividade competitiva entre outras coisas.

*"O **Socialismo** é um sistema sócio-político caracterizado pela apropriação dos meios de produção pela coletividade. Abolida a sua propriedade privada destes meios, todos se tornariam trabalhadores, tomando parte na produção, e as desigualdades sociais tenderiam a ser drasticamente reduzidas uma vez que a produção, sendo social, poderia ser equitativamente distribuída. A proposta de Karl Marx, um dos autores que desenvolveu este tema, é a de que o socialismo fosse um sistema de transição para o comunismo, que eliminaria de forma integral o Estado e as desigualdades sociais." Ver referências*

Como sabemos atualmente o mundo é capitalista, apesar de algumas exceções. Mas que relação isto tem com processo de desenvolvimento de software??

Uma das razões para o capitalismo dar certo é a sua naturalidade, quero dizer com isso que este pensamento/comportamento é intrínseco ao ser humano, todos nós de alguma forma nascemos pensando e agindo assim, uns mais outros menos, e isso acaba refletindo no sucesso que teremos ou não no futuro. Por isso digo que é natural.

*"**Capitalismo** é comumente definido como um sistema de organização de sociedade baseado na*



*propriedade privada dos meios de produção e propriedade intelectual, e na liberdade de contrato sobre estes bens (livre-mercado).*

*“Capitalismo” é o nome que se dá às atitudes econômicas decorrentes naturalmente numa sociedade que respeita a propriedade privada e a liberdade de contrato. As pessoas quando sujeitas a estas condições, com o intuito de satisfazer seus desejos e/ou necessidades, tendem espontaneamente a dirigir seus esforços no sentido de acumular capital, o qual é então usado como moeda de troca a fim de adquirir os serviços e produtos desejados.”* Ver referencias

Quando falamos de socialismo, logo percebemos que ele parece muito perfeito, realmente tudo é pensado em prol de todos, todos são uma peça de um esquema muito maior e que tem um plano ideal para todos.

A desigualdade não existe, porém temos que pagar um preço muito alto por isso, ficamos o tempo todo lutando contra nossos instintos, motivações e tudo mais que move o ser humano em sua busca por uma condição melhor pra si. Temos que sempre pensar no coletivo antes do individual, temos que nos conformar em ter as mesmas coisas que todos, perdemos características que nos tornam únicos em nome de uma causa maior. Isto é muito legal!! Mas é altruísmo demais até para um monge.

Apenas para deixar claro, não tenho intenção nenhuma de discutir ciências políticas ou economia com ninguém, realmente não tenho conhecimento para isso (desconsiderem qualquer bobagem que eu tenha dito, tentem captar a intenção. *risos*).

Minha intenção desde o início é mostrar que os processos e metodologias que conhecemos na vida real como parte do Waterfall não são naturais ao desenvolvimento de software e muito menos à nós desenvolvedores. Eles parecem maravilhosos em um quadro na parede com todo fluxo do PMBOK, por exemplo, mas no dia-a-dia custam muito para serem aplicados e exigem que nademos contra nossos instintos para que cheguemos à algum lugar.

Quando falamos de metodologias ágeis, em primeiro momento parece muito vago, o manifesto ágil em si não se mostra muito técnico, em alguns momentos parece um pouco distante de uma aplicabilidade real. Mas na verdade em sua essência ele tem tudo que nos identificamos. A começar por suas ferramentas, quem na vida nunca se viu praticando pair programming, pois bem isto é uma prática muito útil de uma coisa maior chamada *Extreme Programming*. E não precisamos procurar muito para chegar à conclusão de o *Scrum* tem como consequência uma maior aproximação da equipe e auto conhecimento dentre os participantes, com isso proporciona um maior controle gerencial para quem exercer esta função.

Tudo isso natural para nós programadores e *computeiros*, assim como o capitalismo é para a sociedade e o mercado econômico.

Reforçando, não quero iniciar nenhum tipo de flaming relacionado à política ou economia, quero apenas expôr algumas maluquices que venho pensando ultimamente.

Bom galera, gostaria de dizer que me motivei a escrever essas idéias depois de ler um excelente post do Rodrigo Yoshima no Blog Débito Técnico. É bom saber que ainda existem pessoas que tem a capacidade de provocar o pensamento e instigar a busca por explicações.

*Eu sonho um dia poder trabalhar com uma metodologia ágil, enquanto isso não chega vou me lamentando por aí.*

Abraços, e coloquem suas opiniões! Podem esculachar, *risos*...

Referências:

Socialismo

Capitalismo

Débito Técnico – Não jogue dinheiro com melhoria de processos...

« Entradas anteriores